# 11 | 码控算法：如何控制视频的编码码率？

2021-12-15 李江

《攻克视频技术》 课程介绍 >

**讲述：李江**

时长 17:16 大小 15.82M

你好，我是李江。

上一节课我们讲了带宽预测算法，学习了如何去预测无时无刻不在变化着的网络带宽。准确的预测带宽是实时视频通话技术里面的一个非常重要的环节。

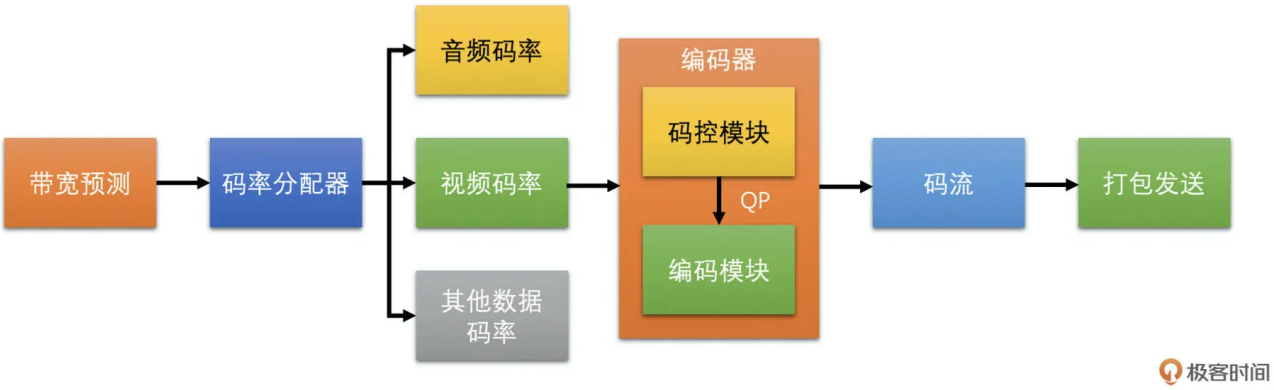
如果不能够很好地预测出实际带宽，那有可能引起数据超发，导致发送数据量大于实际网络的承受能力，继而引起视频画面的延时和卡顿；也有可能预测的带宽太低，导致发送的数据量远低于实际网络的承受能力，不能很好地利用网络带宽，最终导致视频画面模糊和很明显的马赛克现象。因此，一个好的带宽预测算法是至关重要的。

当然，好的带宽预测算法还只是开始，如何在预测出带宽之后能够控制数据的发送码率，使其尽量符合当前的网络带宽也是非常重要的。如果你没有做好发送码率的控制，想发送

多少数据就发送多少数据的话，那跟没有网络带宽预测是一样的效果。要不就画面卡顿，要不就很模糊。

因此，这节课我们就接着来讲讲与网络带宽预估算法一样重要的另外一个算法——视频码率控制算法。我们会先简单看一下码控算法的原理和类型，然后再重点讲解其中最难也是在 RTC 场景中最重要的 CBR 码控算法。我们会非常详细地剖析 CBR 算法的原理，讲解它是如何一步步尽量做到恒定码率的。

通过下面的图你可以清楚地了解码控算法在整个发送端流程中的位置和重要性。




好了，下面我们首先来了解一下码控的原理和基本类型。

## 码控的原理

码控，顾名思义，就是码率控制，它是编码器的一个重要模块，主要的作用就是用算法来控制编码器输出码流的大小。虽然它是编码器的一个非常重要的部分，但是它并不是编码标准的一部分，也就是说，标准并没有给码控设定规则。我们平时用的编码器的码控都是编码器程序自己实现的。

那码控的原理是什么呢？**其实码控就是为每一帧编码图像选择一个合适的 QP 值的过程。**

我们知道当一帧图像的画面确定了之后，画面的复杂度和 QP 值几乎决定了它编码之后的大小。由于编码器无法决定画面的复杂度，因此，**码控的目标就是选择一个合适的 QP 值，以此来控制编码后码流的大小。**当然有些码控算法是可以直接外部指定使用哪个 QP 值去编码的，就不需要编码器的码控算法去做决策了。但是最后的原理是一样的。那接下来我们就来看一下都有哪些码控算法吧。

## 码控的类型

常用的码控算法主要有：VBR（动态码率）、CQP（恒定 QP）、CRF（恒定码率因子）和 CBR（恒定码率）这几种。

## VBR

VBR 指的是编码器输出码率随着原始视频画面复杂度的变化不断的变化。通常当画面复杂或者说运动比较多的时候使用的码率会比较高；而当画面比较简单的时候使用的码率会比较低。VBR 主要的目标是保证视频画面质量，因此比较适合视频点播和短视频场景使用。

## CQP

CQP 很简单就是从头到尾每一个画面都是用同一个 QP 值去编码。根据我们视频编码的课程可知：

> 在画面复杂的时候，残差比较大，相同 QP 值做量化之后的残差还是比较大的，编码之后的图像大小就会比较大。
>
> 而画面简单的时候，残差很小，同一个 QP 值量化之后残差可能很小，甚至都为 0 了，编码之后的大小就会很小。
> 其实，我个人觉得 CQP 是一种特殊的 VBR。但要注意的是 CQP 一般用来衡量编码算法的性能，在实际工程当中不会使用。

## CRF

CRF 是 x264 默认的码控算法。它与 CQP 不同的是它的 QP 是会变化的。在画面运动大的时候，它会根据具体算法提高 QP 值；在画面运动小的时候，它会降低 QP 值。

它的思想是：运动很大的时候，人眼不太关注细节，因此 QP 可以稍微大一点；运动比较小的时候，人眼会将注意力放在细节上面，因此 QP 稍微小一点。所以相比 CQP，CRF 能够更省码率一些。但是 CRF 码控总体上得到的编码后图像的大小，还是随着图像的画面复杂度在变化的。因此，我觉得 CRF 也算是一种特殊的 VBR。

## CBR

另外一种码控算法就是 CBR 了，它是恒定码率的。**这种码控方式用户需要设置一个目标码率值给编码器。**编码器在编码的时候不管图像画面复杂或简单、运动多或运动少的时候，都尽量使得输出的码率接近设置的目标码率。

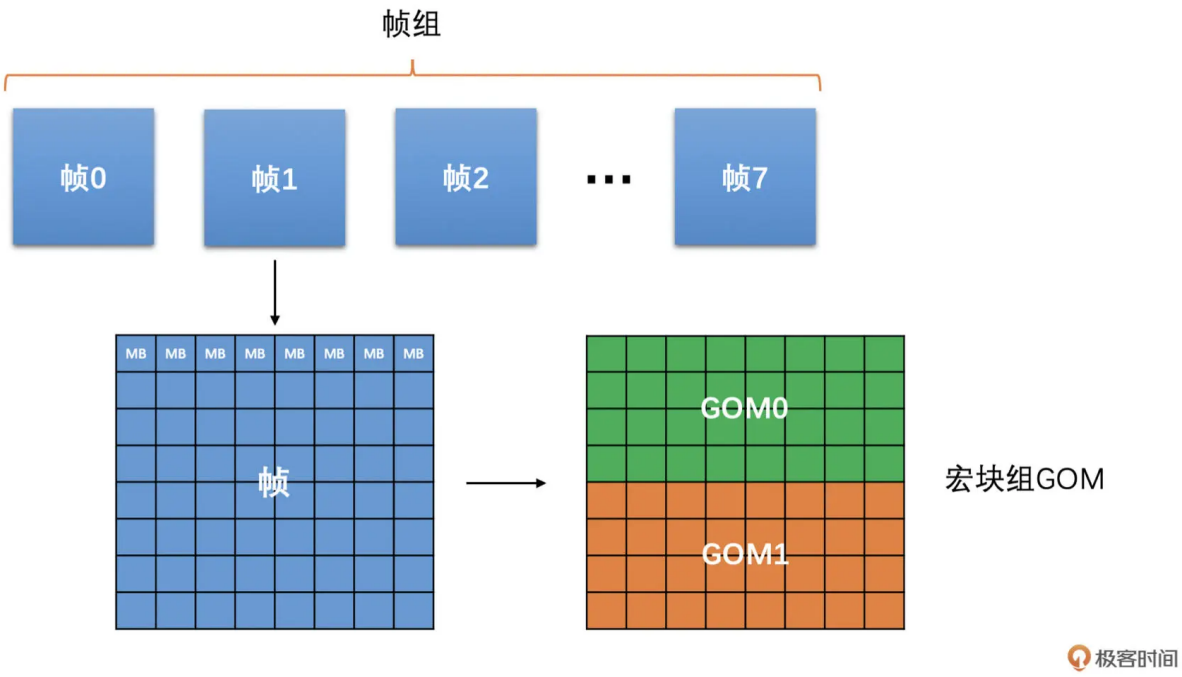
这种方式非常适合 RTC 场景，因为 RTC 场景希望编码的码率跟实际预测的带宽值接近，不能超出目标码率太多，也希望能够尽量有效地利用可用带宽，不能太低于目标码率，从而尽量保证编码后图像画面清晰。

因此，在 RTC 场景中，我们会将预估带宽分出一定比例给视频数据，并将这部分带宽值当作目标码率设置给编码器。需要编码器的码控算法，能够在各种网络状况下和各种画面变化的情况下，都能使得输出的码率尽量接近于当前预估带宽得到的目标码率。

相信你光是看到这个描述就知道非常困难了。所以我们前面说了，CBR 是很重要但也是非常难的一种码控算法。那 CBR 到底怎么做到的呢？我们就来详细讨论一下。

## CBR 算法

其实，为了实现恒定码率，我们需要做很多个步骤，一步步的将输出码率逼近目标码率，而不是一步到位确定 QP 就可以实现恒定码率的目标的。所以，我们会分很多级做调整，分别是帧组级、帧级、宏块组 GOM（Group of MB）级。具体如下图所示：



具体的操作过程如下：

先确定帧组级（帧组就是将连续的几个帧组成一组，一般选择 8 个帧一组）的输出大小尽量接近目标码率。

然后，确定组内的每一帧具体应该分配多少的大小（称之为目标帧大小），才能保证帧组最后输出的大小可以达到要求。

接下来，我们再根据这个目标帧大小，确定一个帧级的 QP 值。

之后，我们再确定帧内的宏块组（宏块组就是连续的几行宏块组成的一组宏块，一般可以选 4 行宏块）应该分配多少大小，来保证当前帧最后的输出大小能接近于目标帧大小。

最后，我们再确定宏块的 QP 值。
还有一个很重要的事情，就是我们需要能够保证在不同的画面复杂度和不同的运动程度的情况下，并且输出码率都要尽量接近目标码率的话，我们还需要先计算得到当前帧的复杂度。

简单来说，这个复杂度是能够大概衡量当前帧在做完预测之后残差值的总体大小的。当然，我们并不是真正去做预测得到残差的，而是通过一些算法近似估算一下残差的大概大小的。因为残差的大小和 QP 值决定了最后图像编码后的大小。

同时，在这里说明一下，因为我们主要讲解 RTC 下的 CBR 码控，所以我们只考虑 I 帧和 P 帧，不考虑 B 帧。等你理解了这些知识之后呢，你再去学习更复杂的 CBR 码控算法就会更轻松一些。

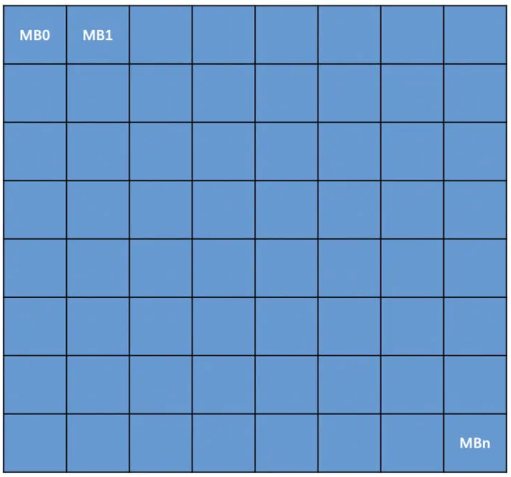
好了，那我们接下来就先讲讲如何计算图像的复杂度，之后我们再依次讨论一下如何在帧组级、帧级、宏块组 GOM 级别做码控操作，最后得到宏块的 QP 值。

## 复杂度求解

根据帧类型复杂度求解可以分为两种算法：第一种就是 I 帧的复杂度计算；第二种就是 P 帧的复杂度计算。

I 帧只做帧内预测，而帧内预测是用编码块周围已编码的像素来预测当前编码块的像素值的。因此，方差是一个比较能够表示 I 帧复杂度的值。

因为方差越大，表示帧的内部变化程度越剧烈，而你用周围的像素去预测当前编码块的像素值的话，有很大的可能会产生较大的残差。而方差越小的话，说明帧内部变化比较小，因此周围像素有较大的概率能够比较好的预测出待编码块的像素值。因此，**我们计算 I 帧的复杂度的时候，是求每一个宏块的方差，最后将帧的所有宏块的方差之和作为帧的复杂度。**具体求解过程如下图所示：



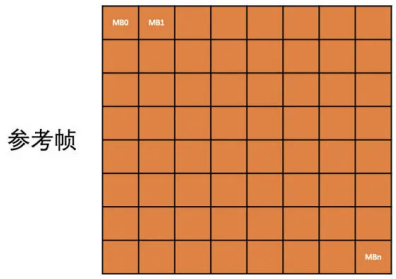

宏块像素的平均值 $= \frac{\text{像素值之和}}{16 * 16}$

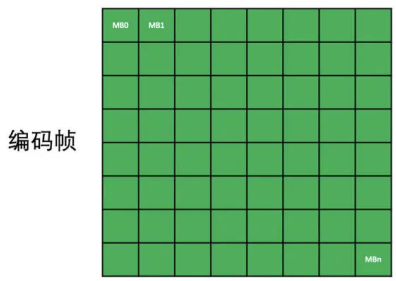
宏块方差 $= \sum_{i=0}^{15} (p(i) - \text{平均值})^2$

I帧的复杂度 = 所有宏块方差之和

极客时间

而 P 帧，主要是做帧间预测。我们知道，帧间预测就是去参考帧中找一个块来作为当前帧编码块的预测块，因此，我们选择使用将当前帧的宏块减去参考帧对应位置的宏块，求 SAD 值，并将所有宏块的 SAD 值加起来作为 P 帧的复杂度。具体求解过程如下图所示：




宏块$MB0$的$SAD$ = (编码帧$MB0$ − 参考帧$MB0$)求像素的绝对值之和

宏块$MB1$的$SAD$ = (编码帧$MB1$ − 参考帧$MB1$)求像素的绝对值之和

宏块$MBn$的$SAD$ = (编码帧$MBn$ − 参考帧$MBn$)求像素的绝对值之和

$P$帧的复杂度 = 所有宏块$SAD$之和

极客时间

当然，我们会保存记录下 I 帧和 P 帧内部每一个宏块的复杂度值，这是因为后面还有地方会使用到。

## 帧组级

CBR 虽然是恒定码率，但它的意思是保证一段时间内的输出码率接近目标码率，比如说 1 秒或者几百毫秒，而不是保证每一帧输出都严格接近目标码率的。

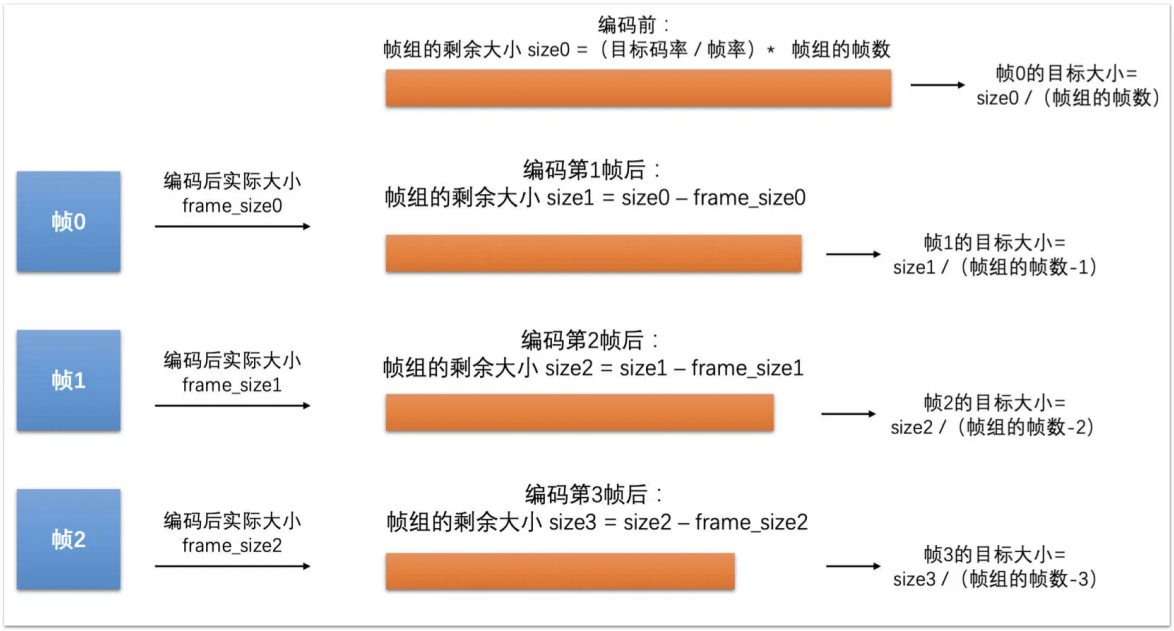
这是因为算法没办法做到每一帧都这么精确。算法是根据一段时间内前面已经编码的结果来调节还未编码帧的 QP，从而来达到一组帧的输出大小尽量接近目标码率的。因此，我们在开始的时候，需要根据目标码率来确定帧组的目标大小，之后再确定帧组内每一帧的目标大小。

我们先根据设定的目标码率和帧率值将两者相除，就可以计算得到每一帧的平均大小。然后我们将帧组的帧数（一般 8 个帧作为一组）乘以帧的平均大小，就是帧组的目标大小了。

在编码器刚开始编码的时候，帧组的剩余大小就是帧组的目标大小。当编码帧组中第一帧的时候，我们将帧组的剩余大小除以帧组的帧数，就得到帧组中第一帧的目标帧大小。当帧组中的第一帧编码完成之后，我们需要用第一帧的实际编码后的大小来更新帧组的剩余大小。

很简单就是将帧组的剩余大小减去第一帧编码后的实际大小。然后，第二帧的目标帧大小就是等于更新后的帧组的剩余大小除以帧组的剩余帧数。随着帧组中的一帧帧不断编码，我们不断更新帧组的剩余大小，不断调整帧的目标大小。

具体计算过程可以参考下图：




你可以很清楚地看到，如果帧组中的前面帧编码后的大小超出平均帧大小的话，后面帧的目标帧大小就会小于平均帧大小，也就是说，前面帧用多了就从后面帧里面扣。同样地，如果前面帧用少了，就补给后面的帧。这样是不是就能保证帧组的最后编码输出码率尽量接近帧组的目标码率了？

举个例子，就像是你一个月有 3000 零花钱，平均每天 100 元。前面 10 天你已经用了 2000 了，那后面 20 天你每天平均只能用 50，要省着点花。如果你前面 10 天只用了 500，那后面 20 天平均每天你可以用 125，可以大方点花。帧组分配帧目标大小也是这个道理。

## 帧级

有了帧组级别码控中计算得到的目标帧大小之后，我们就能够计算当前帧的 SliceQP 了（我们这里为了讲述原理尽量简单清晰，只考虑一帧一个 Slice，多 Slice 原理是一致的，就不展开讲了）。那怎么求呢？

我们根据前面计算得到的当前编码帧的帧复杂度和目标帧大小，再加上前面已经编码完成了的帧的复杂度和编码使用的 QStep（与 QP 一一对应，请参考视频 08 里面的表格）以及使用这个 QStep 编码之后实际的编码大小来计算。公式如下：

$$cmplex_mean_i = 0.5 * cmplex_mean_{i-1} + 0.5 * QStep_{i-1} * frame_size_{i-1}$$

$$frame_complexity_mean_i = 0.5 * frame_complexity_mean_{i-1} + 0.5 * frame_complexity_{i-1}$$

$$QStep_i = \frac{frame_complexity_i}{frame_target_size_i} * \frac{cmplex_mean_i}{frame_complxity_mean_i}$$

$cmplex_mean$是前面已编码帧的$QStep$和实际编码大小的乘积的平均值；
$frame_complexity_mean$是前面已编码帧的复杂度的平均值；
$frame_target_size$是当前编码帧的目标帧大小。

其中 I 帧和 P 帧使用不同的公式，因为复杂度的计算方式不一样。

上面的公式是什么意思呢？其实大体的思想就是：一帧编码后的大小应该是和帧的复杂度成正比的，并且跟帧使用的 QStep 是成反比的。但是具体成多少比例怎么知道呢？

其实呢我们不知道，但是我们可以根据前面已经编码好了的帧估算一下。我们先大体计算一下，它们这些帧的复杂度和 QStep 跟最终的编码大小大概成多少比例。然后再使用这个比例来估算在当前帧的复杂度下，我们大概需要使用多少的 QStep 能使得输出的大小尽量接近目标帧大小。

我们通过上面的公式就计算得到了当前编码帧的 QStep 了，再通过 08 那节课里面的表格就可以转换成相应的 SliceQP 了。

其实，到这里我们就可以用 SliceQP 值去编码每一个宏块了。比如像 VP8 编码，它没有宏块级别的 QP 值，到这里码控就确定了最终 QP 了。但是 H264 还可以在宏块级别调整宏块的 QP，因此，为了更精细化地调节码率，我们还可以根据已经编码宏块的实际使用的大小来调整未编码宏块的 QP。这里就是我们前面提到的宏块组概念了，也就是 GOM。

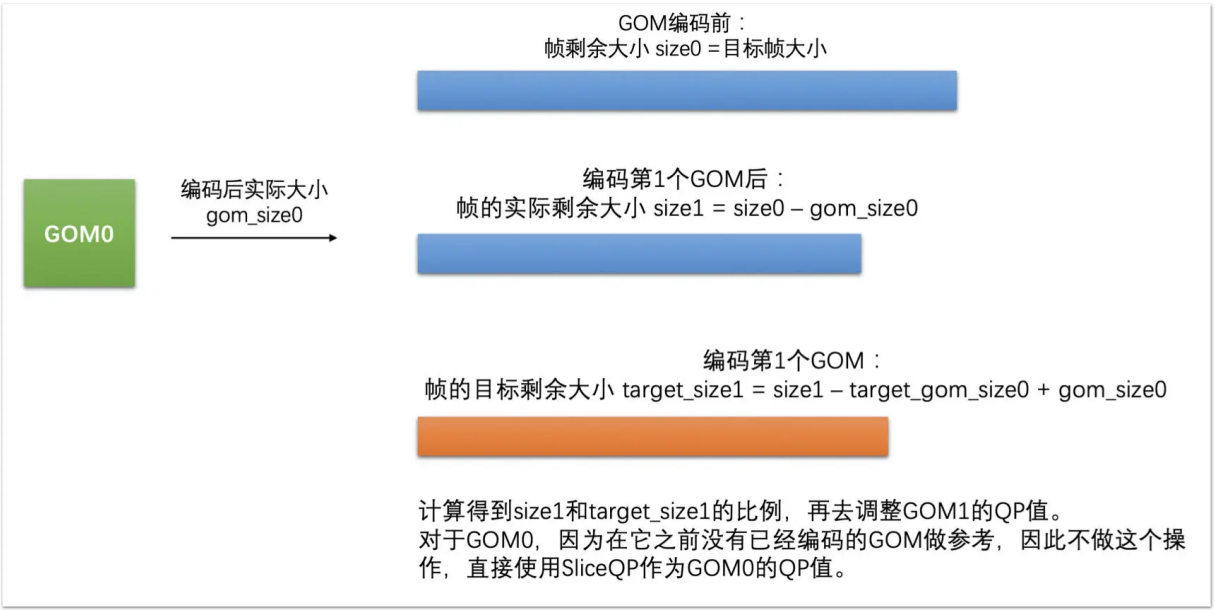
## GOM 级

首先，在开始编码一个 GOM 之前，我们需要计算一下帧的实际剩余大小和帧的目标剩余大小。帧的实际剩余大小是用帧的目标大小减去帧中已编码 GOM 的实际大小。我们再使

用帧的实际剩余大小加上前一个 GOM 的实际编码大小，减去该 GOM 的目标大小，就是帧的目标剩余大小。

这个地方我解释一下，帧的实际剩余大小加上 GOM 的实际编码大小，就是去掉前一个 GOM 的目标大小，再减去前一个 GOM 的目标大小，就是当前的帧目标剩余大小了。

具体计算过程如下图所示：




极客时间

我们将帧的实际剩余大小除以帧的目标剩余大小：

如果这个比例大于 1，说明我们剩余的大小多了，之后的 GOM 可以将 QP 调低一些，我们将后面的 GOM 中的宏块 QP 值减去 1 或者 2 即可；

如果这个比例小于 1，说明我们剩余的大小少了，之后的 GOM 的 QP 需要调高一些，我们将后面的 GOM 中的宏块 QP 值加 1 或者 2 即可。
也就是说，通过这个计算之后，我们就得到了 GOM 中所有宏块的 QP 值了。然后，我们再根据这个 QP 值去编码每一个宏块。

到这里我们还有一个步骤需要做，就是**需要计算一下当前 GOM 的目标大小，以备下一个 GOM 编码的时候做 GOM 级码控计算的时候使用。**

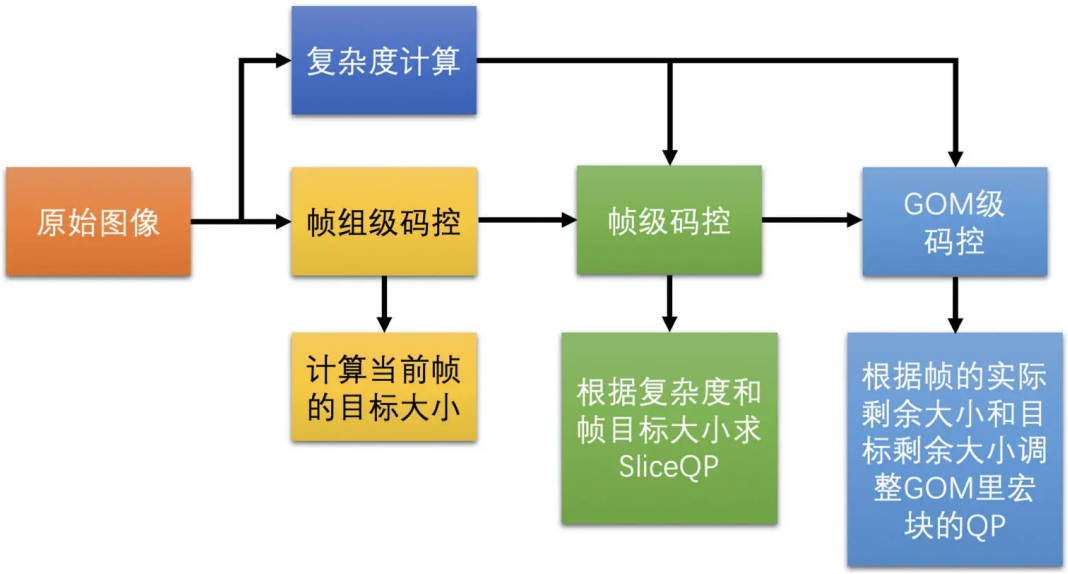
GOM 的目标大小是通过当前 GOM 的复杂度、当前帧剩余 GOM 的复杂度之和以及帧的剩余大小来计算的。计算公式如下所示：

$$GOM的目标大小 = \frac{当前GOM的复杂度}{剩余GOM的复杂度} * 帧的实际剩余大小$$

我们是看当前 GOM 的复杂度占剩余 GOM 总复杂度的比例来分配目标大小的。其中，GOM 的复杂度的值用前面复杂度计算时记录保存的宏块复杂度的值来计算。

其实，我们还可以通过每一个宏块调整一下 QP 的方式来做进一步精细化的调节，但是这个内容有点复杂了，等你学好了这节课之后，我们之后有机会再来深入讲解一下。这里就不展开讲解了。

对于 CBR 码控算法的整体流程，我用下面的图帮你总结了一下，方便你理解和记忆。




极客时间

## 小结

这节课我们主要讨论了码控算法，带你了解了一下码控算法的原理和基本类型。码控主要是为每一帧图像确定 QP 值的过程。如果在图像画面确定的情况下，并且 QP 值确定了的

话，那当前图像编码后的大小就大致确定了，从而编码后的码率大小也基本确定了。

同时，常用的码控算法主要有 CQP、CRF、VBR 和 CBR。并且，我们还对 CBR 进行了深入地探讨。我们知道了 CBR 主要分为：帧组级、帧级和 GOM 级三个级别的调整，并通过一步步不断精细化的调整最后尽量达到恒定码率的目标。

你可以通过下面的图来对这节课加强理解和记忆。

## 思考题

第一个 I 帧和第一个 P 帧的 QP 值怎么确定呢？因为在它们前面没有已经编码好了的 I 帧和 P 帧。

你可以把你的答案和疑惑写下来，分享到留言区，与我一起讨论。我们下节课再见！

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 10 | 带宽预测：3大算法准确预估网络带宽

下一篇 12 | Jitter Buffer：拿什么拯救你，花屏卡顿问题

## 精选留言 (1)

写留言

**晓龙**

2021-12-15

iOS videotool box里只有一种码率设置：ABR：AverageBitRate，这种码控方式是围绕设置的码率上下浮动。从长时间尺度上来看，码率平均值是接近设定的码率的